Advanced Computational Technologies in the GSFC Land Information System

Y. Tian*, C. D. Peters-Lidard†, S. V. Kumar*, J. Geiger‡, S. Olden‡, L. Lighty‡, J. L. Eastman*, P. R. Houser†, J. Sheffield§, E. F. Wood§, K. Mitchell¶, H. Wei¶, G. Gayno¶, P. Dirmeyer‖, B. Doty‖ and J. Adams‖

*UMBC/GEST
NASA, Goddard Space Flight Center
Hydrological Sciences Branch
Greenbelt, MD, USA

†NASA, Goddard Space Flight Center
Hydrological Sciences Branch
Greenbelt, MD, USA

‡NASA, Goddard Space Flight Center
Information Systems Division
Greenbelt, MD, USA

§Dept. Civil & Environ. Engr.
Princeton University
Princeton, NJ, USA

¶NCEP Environ. Modeling Ctr.
NOAA/NWS
Camp Springs, MD, USA

‖Ctr. for Ocean-Land-Atmos. Studies
Calverton, MD, USA

***Abstract*— The Land Information System (LIS) is designed to perform global land surface simulations at down to 1-km resolution in near-realtime. Such an unprecedented scale and intensity pose many challenges in computational technology. In this report we will demonstrate our developments and innovations in high performance computing to meet these challenges and reach our performance goals. These contributions include a fault-tolerant job management system running on a Linux cluster, high-performance, high-availability parallel IO based on GrADS-DODS (GDS) servers with dynamic load-balancing and distributed data storage, and highly scalable data replication with peer-to-peer (P2P) technology. These developments are critical in making LIS a high performance Earth System modeling component with broader applications, and made LIS suitable for production runs in Grid computing environments, and readily interoperable with other modeling components over the Internet.**

I. Introduction

NASA, Goddard Space Flight Center has developed a global Land Information System (LIS) [10], [12] capable of modeling land-atmosphere interactions at spatial resolutions down to 1km. The scientific basis of LIS is an ensemble of land surface models (e.g., CLM [4], Noah [11], VIC [16]) run offline using satellite-based precipitation, radiation and surface parameters, in addition to model-derived surface meteorology. LIS' design also supports the ESTO/CT Round-3 CAN goals of portability, interoperability, and the use of advanced computational technologies.

As reported at the 2003 ESTC, LIS' external interoperability is supported by the adoption of various Earth system modeling standards such as the Earth System Modeling Framework (ESMF) [6]; the PRISM/Assistance for Land Modeling Activities (ALMA) [1] for coupling with other Earth system models, including the Goddard Cumulus Ensemble (GCE) model and the Weather Research and Forecasting (WRF) model. At the 2004 ESTC, we focus on our achievements and innovations in the area of high performance computing and communication, including parallel and fault-tolerant job management, parallel IO and data serving with distributed storage, and highly scalable data replication. All these developments were based on our our 200-node Linux cluster built from commodity components.

The high-resolution land surface simulation of LIS is a highly coarse-grained parallel problem with intensive input/output (IO). To fully take advantage of this feature and satisfy the performance, reliability and resource utilization requirements, we have developed a job management system, which is highly efficient in managing coarse-grained parallel problems on distributed memory systems, especially for Linux clusters. This system features strong fault-tolerance, optimal resource utilization and flexibility. It also integrates distributed visualization so the results can be inspected in real-time without post-processing. In addition, to overcome the IO performance bottleneck, we implemented parallel IO and distributed data storage, and expanded the GrADS-DODS (GDS) [7] client-server system to support its functions over distributed data, and to support GDS server farms with dynamic load balancing. This design increased LIS performance at least 4 times. Moreover, our novel approach in applying peer-to-peer (P2P) technology to high-performance and highly scalable data replication within our Linux cluster, eliminated the performance bottleneck and management complexity of the conventional client–server paradigm. The implementation of the P2P system, BitTorrent (BT) [2], on our cluster enables us to replicate large amount of data to all the cluster nodes simultaneously without traffic congestion, with guaranteed data integrity and with an aggregate throughput at least 5 to 6 times more than conventional NFS based file sharing.

Section II gives a brief introduction to LIS' primary computing platform, the 200-node Linux cluster. The LIS parallel job management system is described in Sec. III, followed by the parallel IO design in Sec. IV. The P2P data replication technology is demonstrated in Sec. V, and a summary of these developments is given in Sec. VI.

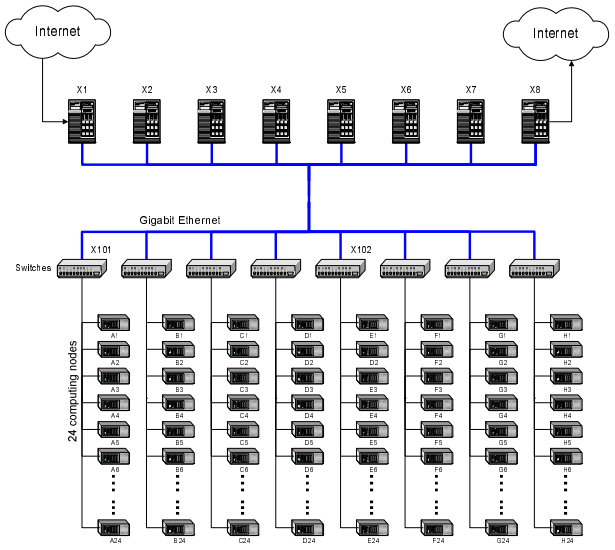

Fig. 1. Physical architecture and network layout of the LIS Linux cluster. The 8 IO nodes' hostnames are X1, X2, ..., X8, respectively, while each compute node is named A1, A2, ..., A24; B1, B2, ..., B24; ..., H1, H2, ..., H24, depending on which of the 8 sub-clusters it belongs.

II. LIS Linux Cluster

LIS' primary computing engine and development platform is the Linux cluster, constructed for the project Milestone-f in the summer of 2002. The cluster was originally built with 200 nodes from commodity components, including eight IO nodes and 192 (24×8) compute nodes.

Each IO node is configured with dual AMD XP 2000 CPUs, 2 GB DDR memory, 220 GB internal disk storage, 2 built-in Ultra-SCSI channels, 2 built-in fast Ethernet adapters, and 1 gigabit Ethernet adapter (two of the IO nodes have two each). In addition, each IO node has one external Promise RAID systems for file storage, with a capacity of 1.2 TB in each RAID system.

Each of the 192 compute nodes has an AMD XP 1800 CPU, 512 MB memory, an internal IDE hard drive of 81 GB, and a built-in fast Ethernet adapter. Subsequent updates added another 512 MB memory to about half of the compute nodes, making the cluster a somewhat heterogeneous system.

Overall, the LIS cluster has 208 AMD XP processors of 1.53 GHz and above, 160 GB of memory, 24 TB of disk space, 192 fast Ethernet connections, and 10 gigabit Ethernet connections. The physical architecture and network layout is shown in Fig. 1. The 192 compute nodes and 8 IO nodes are grouped into 8 branches, with each branch logically having 24 compute nodes and 1 IO node connected to a common Ethernet switch, with all the switches inter-connected. Thus the system can be used as one big cluster with 192 nodes, or 8 small clusters each with 24 nodes, or any combination in between.

III. LIS Job Mangement System

The land surface simulation in LIS is a coarse-grained parallelization problem, in that each grid point on the domain has very weak horizontal interaction with its neighbors. Thus we can divide LIS' global domain into sub-domains, and run each sub-domain independently. Such a problem fits best with a distributed platform such as the LIS cluster and does not require low-latency but expensive interconnects. However, to fully take advantage of this characteristics, we need a job management system which efficiently manages the jobs over the cluster nodes, with strong fault-tolerance and flexibility.

Our job management system is based on the "pool of tasks" scheme. Our implementation distinguishes itself with strong fault-tolerance in the face of node crashes, thus lowering the hardware requirements for the compute nodes. Supported by parallel and distributed input/output, this scheme for LIS has shown very good scalability and optimal resource utilization.

Figure 2 shows LIS' parallelization scheme. The "pool of tasks" paradigm we are using is equivalent to a master–slave programming model, where we use one of the IO nodes as the master node and it distributes jobs to the slave (compute) nodes. We refer to the master node as "farmer" and the compute nodes as "dogs", and the jobs the master node gives out "bones".

The master node ("farmer") will keep three pools on hand when starting the job: pool of unfinished jobs ("bones"), finished ones ("done"), and ones fetched and being processes by compute nodes ("munching"). At the beginning, all the jobs are stored in the "bones" pool. Each compute node ("dog") sends a request to the master to request a job from it, and starts working on it when a job is assigned by the master node. The compute nodes do not get jobs directly by themselves from the "bones" pool, to eliminate the complexity of race condition management. The master node then moves the assigned jobs to the "munching" pool, and starts a timer for each assigned job. The timer specification will be based on the estimation of the execution time a compute node needs to finish a job. When a compute node finishes a job and notifies the master node before the job's corresponding timer runs out, this piece is regarded a finished job, and the master node moves it from the "munching" pool to the "done" pool. And the compute node goes on to request another job until the "bones" pool is empty.

The fault–tolerance is realized with the time–out mechanism based on the timer the master node keeps. If the timer of a fetched job runs out before the compute node reports back, the master node then assumes that that particular compute node must have crashed, and then moves that timed-out job from the "munching" table back to the "bones" table for other compute nodes to fetch. The flow-chart on the left of Fig. 2 shows the master node's job handling and scheduling process, and the various states of the three pools (right) corresponding to the stages in the flow-chart.

This farmer-dog system maximizes resource utilization even when the compute nodes ("dogs") have heterogeneous memory/CPU configuration, as the case with the LIS cluster. Each job ("bone") naturally has different number of total land points, depending on its location on the Earth. The job management system will make sure all the nodes will get to work on the biggest jobs they can handle first, depending on

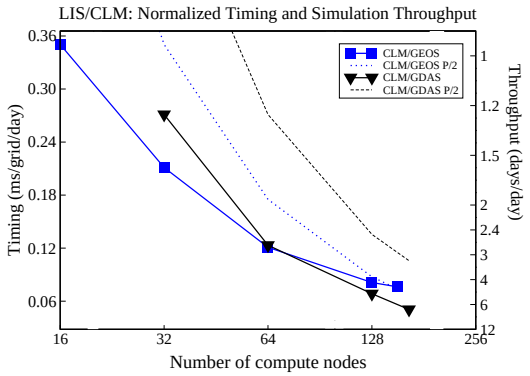

Fig. 3. Normalized timing (left y-axis) and simulation throughput (right y-axis) for LIS with CLM LSM and the two base forcing settings. Five (5) GDS servers were used. The P/2 timing curve is based on 1-node timing extrapolated from 16-node timing for CLM/GEOS, and 32-node timing for CLM/GDAS. The maximum number of compute nodes used for CLM/GEOS is 152, and for CLM/GDAS 164.

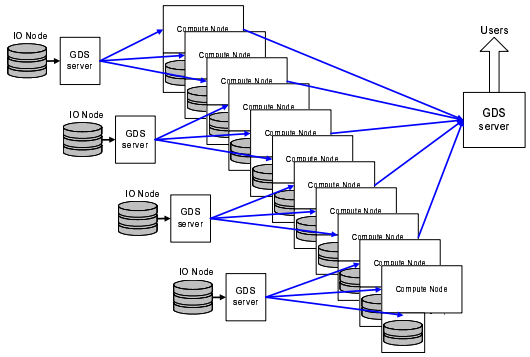

Fig. 4. Parallel IO and distributed storage design for LIS. Input data are fed to the cluster by an array of parallel GDS servers (left). The compute nodes write their output data directly on their local disks. A modified GDS (right) can serve the output data from the compute nodes' disks, eliminating the need of data aggregation onto IO nodes.

their hardware configuration. Only after they finish the biggest jobs can they continue to work on smaller jobs. This scheme ensures the more powerful nodes are not fighting for small jobs with less powerful nodes before they finish the jobs only they can handle. This has proven to be working well on the LIS cluster, since about half of the computer nodes have 1GB memory while the other half have only 512 MB.

Figure 3 shows one of the scaling test results for LIS with the job management system. The results were obtained by timing LIS runs with CLM and either GDAS or GEOS base forcing. For comparison, the estimated P/2 scaling curves are also shown as dashed lines in Fig. 3.

As the figure shows, CLM/GEOS with 16 nodes ran at approximately $0.35ms/grid{\cdot}day$, which is only slightly faster than the target performance requirement of $0.4ms/grid \cdot day$. However, with 32 nodes, CLM/GEOS timed $0.2ms/grid \cdot day$, and CLM/GDAS scored $0.27ms/grid \cdot day$. The timing decreased significantly as the number of compute node was increased, and CLM/GEOS ran at $0.076ms/grid \cdot day$ with 152 nodes, and CLM/GDAS $0.05ms/grid \cdot day$ with 164 nodes, corresponding to throughputs of 4.4 days/day and 6.7 days/day, respectively.

CLM showed better scaling than P/2 curve up to 128 compute nodes. CLM/GDAS still performed better than P/2 even with 164 nodes.

IV. Parallel IO and Distributed Data Storage

As we are using the large number of CPUs for the highly parallel runs for LIS, the input/output quickly became the performance bottleneck. First of all, while LIS is performing global land surface simulations with an ensemble of land surface models (*e.g.*, CLM, NOAH and VIC), it requires a variety of input data, including land surface parameters and model and observational forcing. These input data need to be fed to each individual compute node dynamically with their desired subsets. Next, after getting the input data, each compute node will produce output data with a volume of at least two orders of the input data.

Figure 4 shows LIS' parallel IO design and distributed storage, to solve the IO bottleneck problem. The input data were mirrored on all the IO nodes. For input data serving, we take advantage of GDS server' dynamic sub-setting capability to serve only the subset of the input data to each compute node from request-response transactions, thus reducing the total data traffic. To further boost input data serving throughput, we used multiple GDS servers running in parallel on the IO nodes, and implemented a dynamic load balancing scheme to optimize the collective performance of the GDS servers. To demonstrate how this setting affects LIS performance, we performed tests with 2, 3, 4, 5, and 6 GDS servers. The timing results for NOAH/GEOS runs with 128 compute nodes are shown in Fig. 5.

As the figure shows, the performance improved as the number of GDS servers was increased. With 2 GDS servers, NOAH/GEOS ran at $0.122ms/grid \cdot day$, while with 6 GDS servers, it ran at approximately $0.078ms/grid \cdot day$, with a throughput improvement from 2.8 days/day to 4.4 days/day. However, the most significant performance improvement took place when the number of GDS servers was increased from 2 to 3. More GDS servers helped, but not as much. It is reasonable to expect that more GDS servers (4, 5, and 6) will be more helpful for runs with when the number of compute nodes is further increased.

The output bottleneck poses a much more severe challenge if we follow the conventional approach to first aggregate the output data to a central storage system and then serve the users from there. We completely eliminated the need for central data storage by modifying the GDS server system, to expand its capability to serve data directly off the local disks of every compute nodes. Therefore, each compute node directly writes its output data to its local disk, and then the modified GDS system can treat the collection of the compute node disks as a big disk system, and serve the output data to users from there the same way as from a single local disk. By doing this, the output bottleneck for LIS runs is completely eliminated. Otherwise, it is estimated that it would take 4 to 10 days

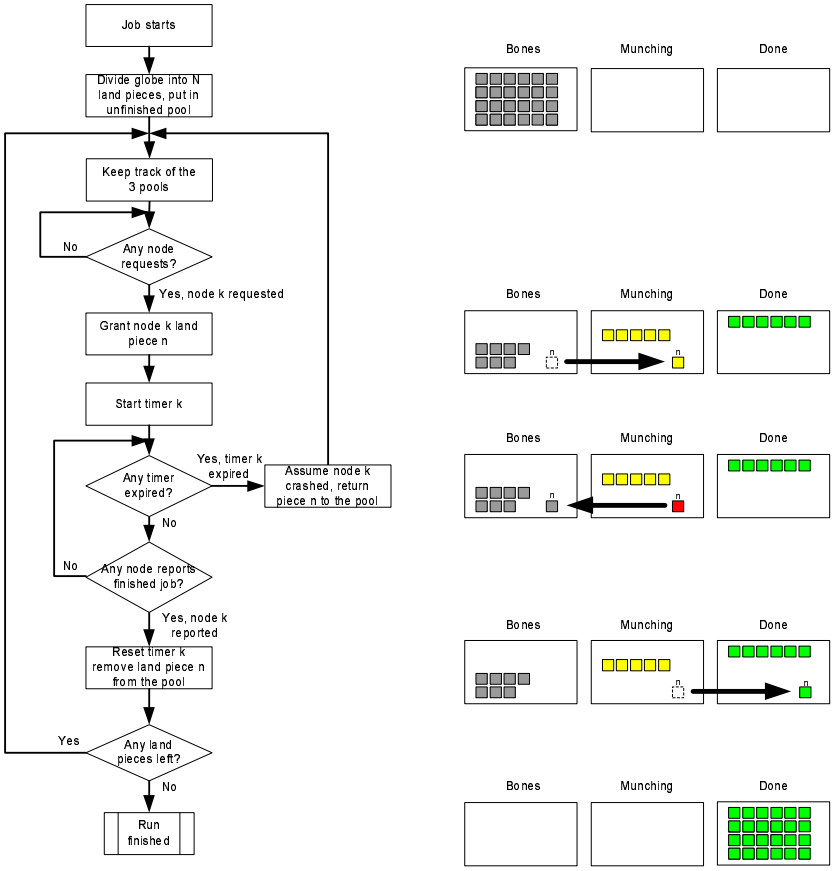

Fig. 2. Task-pool-based parallelization scheme in LIS. Left panel shows the logic of the master node. Right panel shows the movement of jobs in the three pools: bones, munching and done. Each job in the "munching" pool is timed so when a job is timed out (red block), it will be put back to the "bones" pool for other nodes to handle. Time-out happens only when a compute node crashes. On the other hand, code crashes will be detected and handled immediately.

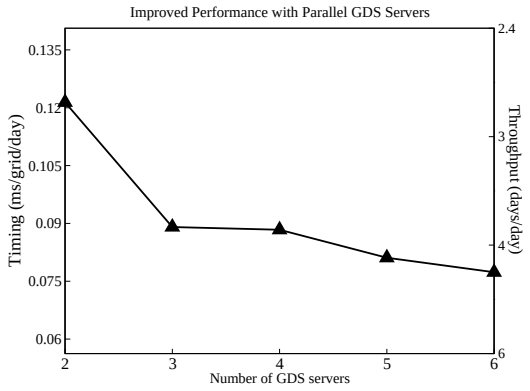

Fig. 5. Normalized timing (left y-axis) and simulation throughput (right y-axis) as functions of the number of GDS servers used, for LIS with NOAH LSM, GEOS base forcing and 128 compute nodes. The output data were saved on each compute node's local disks, and served users the same way as from a central storage.

to finish a 1 day LIS simulation, if the conventional data aggregation and serving were still used.

V. Data Replication with Peer-to-Peer Technology

Although it is trivial to copy small files from one computer to a few other computers over the network via conventional approaches such as NFS, FTP or HTTP based file sharing, it becomes a formidable task to copy a large amount of data from one computer to hundreds of other computers by this means, as with the case for data replication on the LIS Linux cluster. The traditional client–server paradigm, such as NFS, won't scale when tens, or hundreds of clients are copying data from the server, as either the server's bandwidth will be quickly saturated or the server gets overloaded with degraded performance. In addition, the chance of data corruption greatly increases as the data volume and network traffic are increases. Worse still, if a node crashes in the middle of data replication, it has to start over again from the beginning when it comes back online.

LIS needs to pre-stage several Gigabytes of data to every compute node from one node, as these data are static parameters and shared by all the compute nodes, and it would be too expensive to let compute nodes to access the data during runtime over the network. To efficiently replicate these datasets within the cluster without incurring the complexity of client–server approaches, we introduced one of the P2P technologies,

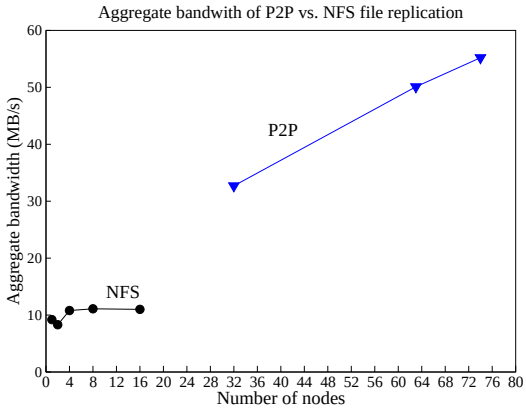

Fig. 6. Comparison of conventional NFS-based and P2P-based data replication performance. The average aggregate bandwidth is defined as the total amount of data transfer divided by the total time taken to finish the transfer. A file containing 2GB random data was used as the test dataset. The tests were performed with all the nodes connected with fast Ethernet links.

BitTorrent (BT), to serve as the data replication channel with the cluster.

To get data from one computer (the "seed") to many others ("peers"), BT first splits the data on the seed into small chunks. Then each peer get a random chunk from the seed. Then the peers can exchange with one another of the chunks and eventually every peer will get all the chunks, thus the whole dataset. Thus the data transfer is mostly taking place between the peers. There is a meta-data server ("tracker") which keeps track which peer has which chunks, and also records the message digest of each chunk. With the message digest, each peer can guarantee the integrity of each chunk, eliminating the possibility of data corruption. Finally, each peer can resume previous transfer without starting over after a crash.

Figure 6 shows the test results on LIS cluster with the conventional NFS based and P2P-based data replication. NFS quickly reached the physical limit of the server's bandwidth with less then 10 clients, and saturated at a bandwidth of 11 MB/s. With BitTorrent, however, the aggregated bandwidth scaled nearly linearly as the number of nodes was increased, and for 74 nodes, it has an aggregated bandwidth 5 times higher than NFS server's. Due to the guaranteed data integrity, no verification was needed after the data were replicated to all the nodes.

VI. Summary

The realtime, high-performance requirement of LIS poses great challenges in the computing technology today. We have to take non-conventional and innovative approaches to meet these challenges and to keep pushing the envelope. Our contribution in the advanced computational technologies made LIS a successful and high-performance Earth modeling system.

We built a low-cost Linux cluster from commodity components to support LIS' intensive computing needs. To fully utilize the computing power of the cluster effectively, and to take advantage of the parallel nature of land surface simulations, we developed our own job management system, which demonstrated great flexibility and strong fault-tolerance, and plays a crucial rule in assuring LIS to run most efficiently and fully utilizing the cluster.

With LIS' scale of computation, the IO is the greatest potential performance bottleneck. We abandoned the traditional approach of output aggregation on central storage devices, and enhanced GDS server with innovative design to let it serve distributedly-stored data to users, and set up the first GDS server farm with parallel servers and dynamic load balancing, to greatly increase the input performance of the whole system.

The need of large scale data replication for LIS' computational needs promoted us to turn to novel approaches. We applied P2P technology to help meet the performance goal of high volume data replication with a large number of nodes. Our implementation of BitTorrent on the Linux cluster proved to be highly effective with our data replication needs.

With the performance results demonstrated by LIS, and with the practical experience we gained from exploring and employing new computational technologies, we believe these technologies will be helpful in other areas as well. In particular, the technologies are not limited to the cluster environment; they will be equally effective in Grid computing environments.

References

[1] ALMA. http://www.lmd.jussieu.fr/ polcher/prismwp3d/.
[2] BitTorrent. http://http://bitconjurer.org/bittorrent/.
[3] F. Chen, K. Mitchell, J. Schaake, Y. Xue, H. Pan, V. Koren, Y. Duan, M. Ek, and A. Betts. Modeling of land-surface evaporation by four schemes and comparison with fife observations. *J. Geophys. Res.*, 101(D3):7251–7268, 1996.
[4] CLM. http://www.cgd.ucar.edu/tss/clm/.
[5] G. J. Collatz, C. Grivet, J. T. Ball, and J. A. Berry. Physiological and environmental regulation of stomatal conducatance: Photosynthesis and transpiration: A model that includes a laminar boundary layer. *Agric. For. Meteorol.*, 5:107–136, 1991.
[6] ESMF. http://www.esmf.ucar.edu.
[7] GrADS-DODS. http://grads.iges.org/grads/gds/.
[8] P. G. Jarvis. The interpretation of leaf water potential and stomatal conductance found in canopies of the field. *Phil. Trans. R. Soc.*, B(273):593–610, 1976.
[9] LDAS. http://ldas.gsfc.nasa.gov.
[10] LIS. http://lis.gsfc.nasa.gov/.
[11] NOAH. ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsm/.
[12] C. D. Peters-Lidard, S. Kumar, Y. Tian, J. L. Eastman, and P. Houser. Global urban-scale land-atmosphere modeling with the land information system. *Symposium on Planning, Nowcasting, and Forecasting in the Urban Zone, 84th AMS Annual Meeting 11-15 January 2004 Seattle, WA, USA.*, 2004.
[13] PVFS. http://www.parl.clemson.edu/pvfs/.
[14] L. A. Richards. Capillary conduction of liquids in porous media. *Physics*, 1:318–333, 1931.
[15] E. Rogers, T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin. Changes to the operational "early" eta analysis/forecast system at the national centers of environmental prediction. *Wea. Forecasting*, 11:391–413, 1996.
[16] VIC. http://hydrology.princeton.edu/research/lis/index.html.